An Introduction to User-Managed Access (UMA)

Eve Maler
VP Innovation & Emerging Technology
eve.maler@forgerock.com
@xmlgrrl

February 9, 2015
Some apps are still in the Web 1.0 dark ages

- Provisioning user data by hand
- Provisioning it by value
- Oversharing
- Lying!
Some other apps are still in the Web 2.0 dark ages

- The “password anti-pattern” – a third party impersonates the user
- It’s a honeypot for shared secrets
- B2B partners are in the “gray market”
Apps using OAuth and OpenID Connect hint at a better, if not perfect, way
What about selective party-to-party sharing?
Our choices: send a private URL...

- Handy but insecure
- Unsuitable for really sensitive data
…or require impersonation…

Import Fidelity Tax Information Into TurboTax®

If you are a Fidelity customer and use TurboTax®, you may be able to import certain information directly from your account into the software. Here’s how.

How to import your information

Once you receive your 1099 statement by mail or through eDelivery, have it available to verify the imported information. Follow these simple steps:

1. Enter your Social Security number (SSN), taxpayer identification number (TIN), or username, and then your password. When asked where to import information from, select Fidelity Investments and enter the same information that you use to log on to Fidelity.com. Then, the tax information available for each of the accounts associated with your SSN should appear.
…or implement a proprietary access management system
Killing – or even **wounding** – the password kills impersonation
IoT 2.0 is here – and it too needs authorization
We have tough requirements for delegated authorization

- Lightweight for developers
- Robustly secure
- Privacy-enhancing
- Internet-scalable
- Multi-party
- Enables end-user convenience
The new Venn of access control

- OpenID Connect
- UMA
- OAuth 2.0

Identity federations
Privacy individuals
Security institutions
UMA protocol standardization in context

- OAuth 1.0, 1.0a
- WRAP
- OAuth 2.0
- Dynamic Client Reg (from UMA/OIDC contributions)
- JWT
- OpenID AB/Connect
- OpenID Connect
- UMA Core, OAuth Resource Set Registration

Timeline:
- 08: OAuth 1.0, 1.0a
- 09: WRAP
- 10: OAuth 2.0
- 11: Dynamic Client Reg (from UMA/OIDC contributions)
- 12: JWT
- OpenID AB/Connect
- OpenID Connect
- UMA Core, OAuth Resource Set Registration

Notes:
- 5 Jan ‘15: 45-day public review of “V1.0 candidate” specs begun: tinyurl.com/umacore & oauthrsr
- Interop test suite development under way
UMA turns online sharing into a Privacy-by-Design solution

The “user” in User-Managed Access (UMA)

Alice’s authorization service hears Bob knocking on the resource – do the rules say he can come in?
UMA-enabled systems can respect policies such as...

Only let my tax preparer with email TP1234@gmail.com and using client app TaxThis access my bank account data if they have authenticated strongly, and not after tax season is over.

Let my health aggregation app, my doctor’s office client app, and the client for my husband’s employer’s insurance plan (which covers me) get access to my wifi-enabled scale API and my fitness wearable API to read the results they generate.

When a person driving a vehicle with an unknown ID comes into contact with my Solar Freakin’ Driveway, alert me and require my access approval.
UMA is about interoperable, RESTful authorization-as-a-service.

Outsources protection to a centralizable authorization server

Has standardized APIs for privacy and "selective sharing"
Use-case scenario domains

Health
Financial
Education
Personal
Government
Media
Enterprise
Web
Mobile
API
IoT
Introducing the OpenUMA community open-source project

OpenUMA

You know that blue “Share” button in Google Apps? Ever wanted to add a feature like that to your own app or API ecosystem? The UMA protocol enables you to do just that.

User-Managed Access (UMA) is an OAuth-based protocol that enables an individual to control the authorization of data sharing and service access made by others.

The OpenUMA community shares an interest in informing, improving, and extending the development of UMA-compatible open-source software as part of ForgeRock’s Open Identity Stack. Currently no open-source OpenUMA code has yet been published, but keep an eye out in early 2015!

On this page:

- About the UMA Standard
- Project goals
- Sample Use Case
- Infographic: UMA
- The OpenUMA video
- OpenUMA blog posts
- Get involved!

Leaderboard

#1 Peter Major 405
#2 Victor Ake 398
#3 Brad Tumy 350
#4 Scott Heger 342
#5 David G. Simmons 341
#14 Eve Maler 147

The leaderboard is based on our rockin’ informal points system, read about it here.

Recently Active Members
Share with others

People

bob@gmail.com  

Basic Authentication

✓ Basic Authentication

Two-step Authentication

Done
Under the hood, it’s “OAuth++”

Loosely coupled to enable an AS to onboard multiple RS’s, residing in any security domains.

This concept is new, to enable party-to-party sharing driven by RO policy vs. run-time consent.

asynchronous consent by RO drives RqP’s access through data associated with RPT.
The RS exposes whatever value-add API it wants, protected by an AS.

The RPT is the main "access token" and (by default – it's profileable) is associated with time-limited, scoped permissions.

The RPT is a tuple of these four entities; it may potentially span ROs because the C or RqP should not know which RO controls which resource.
The AS exposes an UMA-standardized protection API to the RS

- Resource registration endpoint
- Permission registration endpoint
- Token introspection endpoint

The PAT protects the API and binds the RO, RS, and AS

- Authorization API
- Resource server
- RqP requesting party
- Client
- AS authorization service
- RS resource service
- RO resource owner

The PAT drives RqP’s access through data associated with RPT
The AS exposes an UMA-standardized authorization API to the client

- **RPT endpoint**
- **RO resource owner**
- **AS authorization API**
- **C client**
- **RS resource server**
- **PAT protection API token**
- **RqP requesting party**
- **AAT authorization API token**

The AAT protects the API and binds the RqP, client, and AS

The client may be told: “need_info", necessitating trust elevation for authentication or CBAC (or, through extension, ABAC)
The AS can collect requesting party claims to assess policy

A “claims-aware” client can proactively push an OpenID Connect ID token, a SAML assertion, a SCIM record, or other available user data to the AS per the access federation’s trust framework.

A “claims-unaware” client can, at minimum, redirect the requesting party to the AS to log in, press an “I Agree” button, fill in a form, follow a NASCAR for federated login, etc.
UMA Binding Obligations

- Distributed authorization across domains? Scary!
- This “legal” spec enables parties operating and using software entities (and devices) to distribute rights and obligations fairly in access federation trust frameworks.

Important state changes when new pairwise obligations tend to appear:
- Token issuance
- Token status checks
- Permission registration
- Claims gathering
- Access requests
- Successful access
Thank you!

Eve Maler
VP Innovation & Emerging Technology
eve.maler@forgerock.com
@xmlgrrl